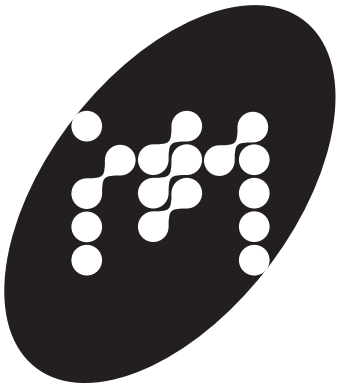

MINEMA

Survey on Position-Based Routing[1]

Filipe Araújo and Luís Rodrigues
Universidade de Lisboa
{filipius,ler}@di.fc.ul.pt

Abstract

In this paper we review position-based routing for wireless *ad hoc* and for wired networks. This form of routing assumes that nodes are aware of *i*) their own position; *ii*) the position of their neighbors (obtained through the exchange of beacon messages); and *iii*) the position of the node that is the intended destination of the packet.

Document Identifier	TR-01
Document Status	Published
Created	13 October 2005
Revised	24 January 2006
Distribution	Public

[1] This paper is based on a chapter of the submtitted PhD Thesis of Filipe Araújo [1]

Contents

1 Overview

The goal of routing is to deliver a packet from a source node S to destination node D in a network of nodes (which we can represent as a graph). To solve the routing problem, nodes of the network execute a distributed "routing scheme". A routing scheme is comprised of two parts [see, for instance, 2]: *i*) a distributed algorithm, here known as the routing algorithm, running at every node, which is responsible for determining the output port (i.e., the next hop) of a packet; and *ii*) a pre-processing algorithm that, given the initial connection graph G, must create whatever information is necessary to the routing algorithm (e.g., routing tables or a subgraph of G). In the worst case, optimal routing schemes may require as much as $O(n \log n)$ memory space at each node. For this reason, there are many "compact routing schemes", which try to reduce these requirements, for instance, by re-arranging the identification of nodes [3]. Position-based routing can be seen as a form of compact routing in which nodes receive identifications that depend on their geographical positions. Usually, in literature, authors assume that the identification of a node is precisely the position that it occupies in the $\mathbb{R}^2$ plane. Since a packet includes the geographical position of the destination, nodes will know in which direction to forward it. On the contrary, in IP networks, nodes perform routing using protocols that derive from Dijkstra [4] or Bellman-Ford algorithms [5, 6].

1.1 Advantages of Position-Based Routing

For the sake of scalability, we will focus on single-path position-based routing schemes, i.e., we do not focus on algorithms where nodes create additional instances of the packet they are forwarding. Additionally, we assume that position of nodes encodes topological information, i.e., in general, nodes that are geographically close are also topologically close. This assumption allows nodes to avoid extensive topology updates, thus saving precious resources. This is especially important in fast changing networks, like wireless *ad hoc* networks or peer-to-peer networks. For instance, in wireless *ad hoc* networks, it is difficult to determine the underlying topology for two basic reasons: *i*) it may change too fast, thus generating too many global update messages that will consume available battery power and bandwidth; and *ii*) nodes have limited memory and usually they will not be able to store all the topology even if they could collect it. According to several experimental work [e.g. 7, 8], routing schemes that do not use positional information and that are based on the exchange of routing tables, like DSDV [9], AODV [10] and DSR [11], do not scale. Additionally, [7] showed that routing table size grows linearly for the DSDV algorithm, while it grows logarithmically for a comparable position-based routing scheme [see also 12]. These limitations may also affect wired nodes, namely in peer-to-peer networks, where topology also tends to change very fast.

2 Assumptions

Position-based routing stands on top of a number of assumptions, including the following. The most important one is that nodes can determine their own position. To get positional information, nodes can use a Global Positioning System (GPS) receiver, if they are outdoors[2]. In alternative, wireless nodes can use techniques based on signal strength information, available in the standard IEEE 802.11 technology [15, 16]. For wired nodes there are also some attempts to provide a mapping service capable of returning a position given the IP address of a node [17]. A second assumption is that nodes can determine location of their neighbors. This usually implies the exchange of a small number of packets between neighboring nodes to make their own positions available to others. Final assumption states that nodes can determine the position of the destination. Reasonability of this assumption depends on the concrete network. Usually, the problem of

[2] A completely different approach is followed in [13] and [14] that use logical instead of geographical positions.

determining the network address of the destination is separated from the routing problem (take the Domain Name Service, DNS, for example). However, in practice, it may be difficult to use a different layer to provide a location service atop of, for instance, a wireless network. For this reason, there are solutions that integrate the routing with the location problem, e.g., the Grid Location Service [8]. In the case of position-based distributed hash tables (DHT), e.g. [18], this service can be provided by the DHT itself and therefore, the assumption is perfectly reasonable. Henceforth, we focus on the problem of sending a packet from source node S to destination node D, considering that both of them have locations known to S.

3 Definitions

3.1 Notation

We will use the following conventions for notation:

- nodes (also designated as vertices) will be represented with capital letters, for instance, node A;
- the set of all nodes of the graph is designated as V;
- edges are represented by the two nodes that define them, for instance, node A and node B may define edge AB;
- distance between two nodes A and B will be represented by $\|AB\|$;
- a triangle defined by nodes A, B and C is represented by $\triangle ABC$;
- the circumcircle of $\triangle ABC$ is represented as $\bigcirc ABC$;
- the circle whose diameter is defined by two nodes A and B is represented by $d(A, B)$;
- the circle centered at A with ray r is represented as $b(A, r)$ (ball centered at A with ray r);
- an angle between line segments AB and AC defined at A is interchangeably represented by $\angle BAC$ or $\angle CAB$ — the vertex where the angle is measured stays in the middle, while the position of remaining vertices is arbitrary. Unless stated otherwise, this angle is always $< \pi$.

Additionally, we will introduce in the following sections a number of models and definitions which are necessary to understand existing position-based pre-processing and routing algorithms.

3.2 Unit Disk Graph

Given the set of wireless nodes V, the Unit Disk Graph model (UDG) assumes that i) communication range of nodes is a perfect circle in $\mathbb{R}^2$; and ii) all nodes have the same communication range. Hence, the Unit Disk Graph of V, $UDG(V)$, is comprised of all edges not longer than maximum communication range between all pairs of nodes. Assuming that communication range is 1, graph $UDG(V)$ includes *all* possible edges whose length is at most 1 (also known as short edges as opposed to long edges which are longer than 1).

Given this definition, every node Q that is inside $b(P, 1)$ is a neighbor of P, i.e., $Q \in N(P)$. Alternatively, Q is a 1-hop neighbor of P. If $Q \in N(P) \vee \exists C \in V : C \in N(P) \wedge C \in N(Q)$, Q is said to be a 2-hop neighbor of P, or simply 2-neighbor of P, and vice-versa. This definition can be extended to define neighborhoods between nodes that are separated by an arbitrary number of hops. Hence, two nodes that can reach each other in k or fewer hops are considered to be k-neighbors.

Henceforth, we will usually assume that we are using the UDG model. This model is more adequate to describe wireless *ad hoc* networks than to describe wired networks. Hence, the algorithms that depend on this model can only be used in wireless environments. This is especially important in the case of the pre-processing algorithms, which we present in Sections 5 and 6. Given some inherent limitations of this model [19, 20], there is ongoing work to generalize position-based routing to more realistic settings. One of these is presented in the next section. Other authors, like Kim *et al.* in [21] try to eliminate any restrictions on the connectivity of the nodes, while, for instance Lim *et al.* [22] try to mitigate the uncertainty on the estimation of location.

3.3 The $R/r \leq \sqrt{2}$ Model

Barrière *et al.* in [23] made a better attempt to model real radio transmissions of wireless nodes. Authors consider two circles for node transmission capacity: a smaller circle with ray r and a larger circle with ray R, which correspond, respectively, to the minimum and maximum admissible transmission range. This means that messages broadcast by some node must reach all other nodes at a distance of up to r and may or may not reach nodes that are at a distance of up to R — i.e., transmission range does not need to be a circle, but must be within two circles. For the reasons that we will see in Section 5.1, the ratio R/r must be at most $\sqrt{2}$. Although more robust, this model is not used very often in literature. Therefore, we will generally restrict ourselves to algorithms and techniques that assume the UDG model.

3.4 Localized Routing Scheme

To conserve resources, one often requires routing schemes to be localized. A routing scheme is localized if there is a constant n such that a node only uses i) its own information, ii) information of neighbors that can be reached in up to n hops and iii) information of a constant k number of additional nodes [see 24]. A routing scheme is said to be n-localized if n is the smallest constant that satisfies the above condition. Occasionally, to simplify, we refer to "localized routing algorithms", but what we really mean is that the pre-processing algorithm and the distributed routing algorithm are both localized (usually, only the pre-processing algorithm could be non-localized, because the routing algorithm uses information local to the node).

3.5 Spanning-ratio

Although creating economical routing schemes is very important, ensuring good performance is not less important. Hence, one criterion used to evaluate the pre-processing step of a routing scheme is the *quality* of the subgraph created. We will call G to some initial graph, which we will often assume to be UDG in a wireless network, and H to the subgraph created by the pre-processing algorithm. As we will show ahead, this subgraph H allows the routing algorithm to converge. H is said to be a "t-spanner of G" if and only if:

$$\underset{\forall S,D \in V}{max} \left\{ \frac{\| \Pi H(S,D) \|}{\| \Pi G(S,D) \|} \right\} \leq t$$

This means that for all nodes S and D, shortest path between S and D in H, $\Pi H(S,D)$, is at most t times longer than in G, $\Pi G(S,D)$. t is known as the "length stretch factor". In a sense this factor indicates the quality of the subgraph. The smaller it is, the better the subgraph is.

When the graph G is the complete Euclidean graph determined by V, the above expression defines an "Euclidean t-spanner".

3.6 Competitive-ratio

The reader should notice that the spanning-ratio of a subgraph is only a bound to the performance of a routing scheme. We still need to develop routing schemes that select paths which are *close* to the shortest path. The "competitive-ratio" is used as an accurate measure of the quality of the routing scheme (RS) and is defined as follows:

$$\text{competitive-ratio(RS)} = \max_{\forall S,D \in V} \left\{ \frac{\| AG(S,D) \|}{\| \Pi G(S,D) \|} \right\}$$

$\| AG(S,D) \|$ is the length of the shortest path between S and D, found by the routing scheme A in graph G; $\| \Pi G(S,D) \|$ is the length of the shortest path, between the same pair of nodes, existing in G. A routing scheme is said to be "t-competitive" if its competitive-ratio is t. Again, this is a worst-case definition, because the competitive-ratio is determined by the pair of nodes for which results are worst. Often, the name "stretch factor" is also used instead of "competitive-ratio".

One interesting known fact is that no localized routing scheme can be t-competitive for any constant t. Kuhn *et al.* [25] showed that if c is the cost of the best path for a given pair of nodes, the cost for any localized position-based routing scheme can grow to $\Omega(c^2)$. More precisely, there are graphs where any deterministic (randomized) position-based routing scheme has a (expected) cost of $\Omega(c^2)$. Furthermore, this applies to the number of links traversed, to Euclidean distance or to energy spent transmitting the packet. To understand the reason for this, the reader should refer to Figure 1, which shows a variation of the "lower bound graph". The dots represent the nodes, while the lines represent the links between the nodes. In a wireless network, the distance between nodes of the inner circumference is precisely 1. Therefore, these nodes are connected. However, the other nodes of the radial lines can only communicate with the immediate neighbors in their own radial, because the distance to nodes of other radials is greater than 1. Another key aspect of this graph is that there is only one radial that gives access to the outer circumference. Having access to local information only, a node cannot know the topology and, as a consequence, it cannot know which of the radials connects to this circumference. This means that in the average a packet needs to try half the radials before it gets to the right one. Hence, if the shortest path (e.g., in number of links) to some node in the outer circumference is c, a localized routing scheme may end up using $\Omega(c^2)$ links to reach the only node that gives way to that outer circumference.

3.7 Delaunay Triangulation

The *Delaunay triangulation* (DT) of a node set V, also represented as $Del(V)$, is the set of edges satisfying the "empty circle" property: edge AB belongs to the triangulation if and only if there is a circle containing A and B, but not containing any other node. An important property of $Del(V)$ states that the circumcircle of a triangle does not contain any node of V. To this property we call the "empty circumcircle" property. The DT has an associated dual concept called the "Voronoi tessellation". The Voronoi tessellation partitions the space into convex polytopes in the following way. Given a node set V, the polytope of node N is comprised of the points that are closer to N than to any other node of V. In this paper we will restrict ourselves to two-dimensional spaces. Therefore, we call simply "cell" to the Voronoi polygon of node N. Figure 2 shows the relation between the Voronoi tessellation (dashed lines) and the Delaunay triangulation (solid lines): two nodes share a Delaunay edge if and only if their Voronoi cells have a common border. We can also see the empty circumcircle property for two triangles, as no fourth node is inside the depicted circumcircles.

Although Delaunay triangulations have many other applications (e.g. Computer Graphics), here, we are interested in their advantages for routing purposes [26, 27, 28, 29, 30].

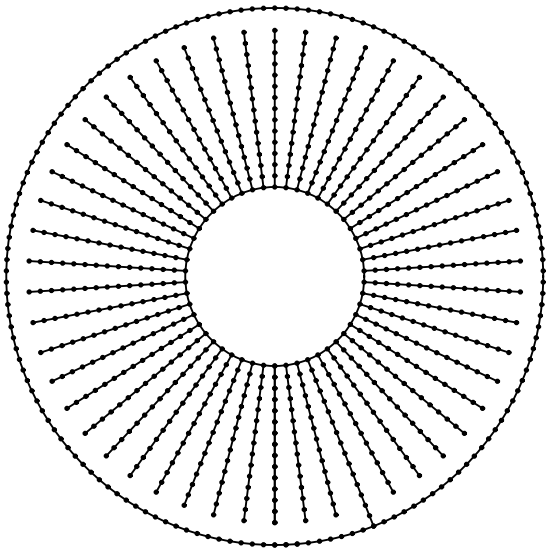

Figure 1: Variation of the lower bound graph

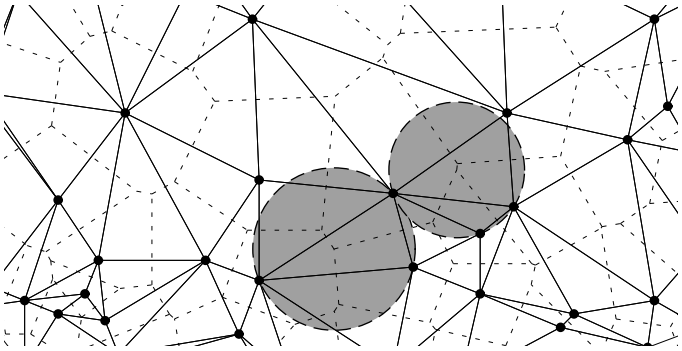

Figure 2: Delaunay Triangulation, Voronoi tessellation and the empty circumcircle

4 Routing Algorithms

In this Section we briefly overview some of the most important position-based routing algorithms. All of these algorithms assume that nodes have only a partial view of the graph, such that they must relay packets to the neighbors, to reach a distant and unseen destination. At this moment we do not care about the pre-processing algorithm that nodes use. We postpone this issue to the following sections, as the pre-processing algorithms are strongly dependent of the kind of network that nodes use, either wired or wireless.

4.1 Basic Position-Based Routing Algorithms

In the following descriptions we assume that the algorithms are executed at node S, packets are destined to node D and there is a graph that determines the neighbors of node S. This graph must be created by a pre-processing algorithm, before the node starts forwarding packets. In the case of wireless networks, this resulting graph is typically a subgraph of UDG.

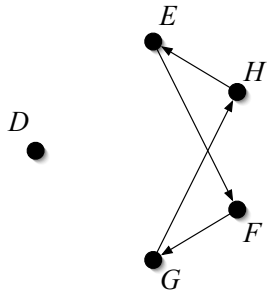

Figure 3: Compass routing can create a loop between nodes E, F, G and H

Greedy

Greedy routing algorithm [31] is a memoryless algorithm (only requires information about destination). When using greedy forwarding, a node selects for the next hop, the node that is closest to destination (including itself). It is easy to come up with examples where this algorithm does not converge, due to local minima that occur in regions void of neighbors.

Compass

Consider the angle formed by line segments SN and SD, where S is the forwarding node, N is a potential next hop and D is the destination. The compass routing algorithm [24], forwards packets to the neighbor N that forms the smallest angle $\angle NSD$ with the destination. Compass routing algorithm is also memoryless.

In [32], we can find Figure 3, which shows an example where the compass routing algorithm may fall in a loop, between nodes E, F, G and H. D is not aware of any of the other nodes (and vice-versa); E and G are also not aware of each other. Hence, the neighbors of E are F and H. Consider that the packet is at node E. F is the node that minimizes the angle with D at E, i.e., $\angle FED$. Then, F forwards the packet to G, which, in turn, forwards the packet to H, which sends it back to E, thus forming a loop. Hence, the compass routing algorithm is not loop-free.

Randomized Compass

Randomized Compass routing algorithm [26] is a variation of the compass algorithm that avoids loops with random decisions. It is also a memoryless algorithm. Consider as in the compass routing the line defined by forwarding node and destination. At each node, two options are considered to route a packet: the neighbor with smallest angle above that line and the neighbor with smallest angle below that line. One of those neighbors is randomly chosen to be the next hop.

Most Forwarding within Radius

Consider that line SD is the x-axis, where S is the forwarding node and D is the destination node. In the Most Forwarding within Radius [33] (MFR) node S forwards the packet to the node A that maximizes progress along x-axis. A is therefore the node that minimizes the dot product $\overrightarrow{DS} \cdot \overrightarrow{DA}$, assuming that it is positive (if it is negative $\angle SDA > \pi/2$, which means that S and D must be neighbors). This algorithm is also loop-free and memoryless.

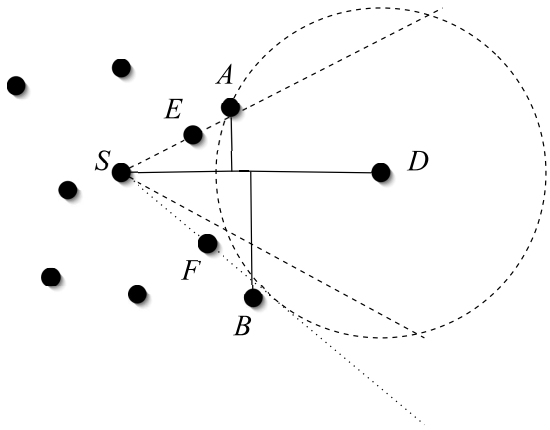

Figure 4: Next hops of several routing algorithms

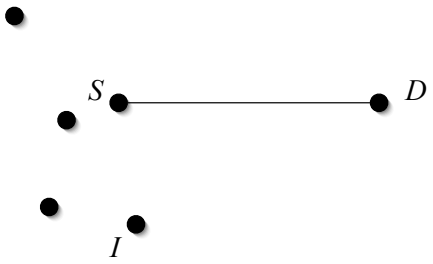

Figure 5: Next hop of GEDIR

Geographical Distance Routing

GEographical DIstance Routing [32] (GEDIR) resembles greedy algorithm, with a subtle difference. Packets are sent to the neighbor A that is closest to destination D, despite the distance of the current node, S, to the destination. This means that a packet can be sent to some node A that is actually more distant from D than the sending node S. The rationale for this is that A may have some neighbor that is closer to D than S is. The only kind of loop that may occur in this algorithm is between two consecutive nodes and, therefore, one can make it loop-free [32].

Figures 4 and 5 try to illustrate the routing algorithms described before. Forwarding node S is aware of all the nodes depicted except of destination D. Greedy algorithm will choose node A, as this node is closest to destination. Compass will choose E as this has the smallest angle, while randomized compass also allows the selection of F, as F defines the smallest angle in the opposite side. MFR will select B. Finally, GEDIR (Figure 5) attempts to deliver the packet even when S is a local minimum. In this case, the next hop is node I, no matter if $||ID|| < ||SD||$ or not.

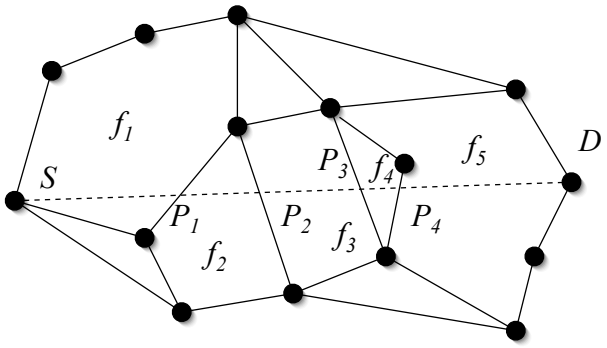

Figure 6: Sequence of faces using FACE-1 routing algorithm

4.2 Right-Hand Routing Algorithms

None of the deterministic algorithms presented in Section 4.1 can ensure routing convergence. For each one of these algorithms, there is always a graph where they will fail to find a path to destination, even when that path exists. However, it is unacceptable to successively fail to find a path if such path exists. Therefore, they are not adequate to environments where the shape of the graphs is not controlled and is pretty much arbitrary. Fortunately, there are algorithms that can overcome this problem. This is the case of the algorithms based on the right-hand rule [34]. This rule states that all the walls of a maze can be visited if the visitor never lifts his/her right-hand of the wall. If instead of a maze we have faces of a connected planar graph (i.e., without intersecting edges), such rule allows a packet to visit all the edges of a face. This means that a packet routed under the right-hand rule always returns to the source node. Algorithms based on the right-hand rule require $O(1)$ memory, because packets need the sender information, S, to determine which edges intersect line segment SD. Next, we present two algorithms that are based on the right-hand rule. The key point here is that these algorithms must be executed on planar graphs, otherwise they may fail to converge. Ahead in the text, we will focus on the problem of making a graph planar.

FACE-1

A very simple routing algorithm based on the right-hand rule is Compass II [24], later renamed as FACE-1 in [35]. In this algorithm, nodes forward the packet from face to face, always getting closer to destination. The packet goes through faces that intersect line segment SD (known as r in the Algorithm 1). Assume that the packet is inside face f. We set the variable P to the initial source node S. Then, packet goes from edge to edge (either clockwise or counterclockwise) until it reaches some edge e that intersects r. The packet must keep track of which edge of face f intersects r closest to destination D. Then, when the packet returns to that edge it is certain that there is no other intersecting point P closer to D. At this point the packet may switch from face f to the next face, closer to D. Algorithm 1 describes FACE-1. Figure 6 exemplifies the use of FACE-1 in a graph with 5 different faces, fi from S to D. Points Pi are the intersections of the edges of the face with line segment SD. The forwarded packet will cross faces in ascending order $f1, f2, \ldots, f5$ and the same with the intersecting points $P1, P2, P3, P4$,

To see why FACE-1 needs a planar graph, Figure 7 shows a case where FACE-1 fails to converge in the presence of intersecting edges. Here, the packet would make a cycle like $S-A-B-C-A-S$ without ever finding an edge intersecting SD closer to D than S is. Most algorithms based on the right hand rule will also fail to converge in this case.

Algorithm 1 Algorithm FACE-1

```
constant S ← source
constant D ← destination
constant r ← line segment SD
P ← S
while P ≠ D do
   f ← first face that intersects line segment PD from P to D
   for all edges e of f do
      if e intersects r at point X and X is closer to D than P then
         P ← X
      end if
   end for
   traverse f again until reaching edge where P was found
end while
```

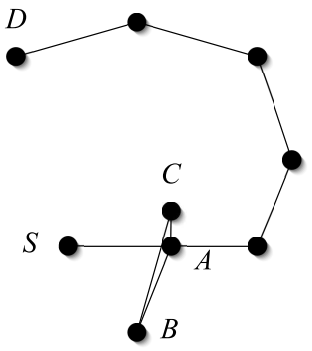

Figure 7: Packet from S will never reach D

FACE-2

A slightly different version of this algorithm, FACE-2, was proposed in [35] (see Algorithm 2 executed at graph G). FACE-2 tries to avoid the multiple traversals of the same face that may severely affect performance of FACE-1. FACE-2 switches face whenever the packet reaches a new intersection between r and a different edge, which is closer to destination. Then, the process is repeated for the new face f. Note that f may actually be the same face, if f is not convex. While FACE-1 reaches destination node D in at most $3|E|$ hops, where $|E|$ is the number of edges of graph, FACE-2 has a worse theoretical upper bound, but works better in practice. The experimental results of [35] show that, in spite behaving really badly for some input graphs, FACE-2 generally outperforms FACE-1. Nevertheless, neither one of these algorithms may be used as a standalone routing method, due to their bad performance. As we discuss next, we use these algorithms in complement with better performer, but unreliable algorithms, like greedy or compass, for instance.

4.3 Hybrid Position-Based Routing Algorithms

Greedy Perimeter Stateless Routing

One algorithm that explores the duality between efficient and reliable approaches is the Greedy Perimeter Stateless Routing, GPSR [36]. GPSR is based on the original idea of Bose *et al.* [35]. GPSR is a well-known and fairly simple routing protocol for planar graphs. It uses the greedy routing algorithm and the right-hand rule, which is called perimeter routing in this context. The main idea is to use greedy algorithm whenever possible and switch to perimeter routing whenever

Algorithm 2 Algorithm FACE-2

```
constant S ← source
constant D ← destination
P ← S
while P ≠ D do
   f ← face of G with P on its boundary that intersects PD
   traverse f until reaching an edge UV that intersects PD at some point P' ≠ P
   P ← P'
end while
```

greedy gets stuck at a local minimum. When in perimeter mode, GPSR is similar to the FACE-2 algorithm. Then, routing proceeds using perimeter routing and switches back to greedy as soon as it finds some node closer to destination than the previous minimum. The intuition is that greedy algorithm performs better, but it is unreliable, while perimeter algorithm always works if the underlying graph is planar.

AFR and GOAFR$^+$

Kuhn *et al.* [25] have taken the idea of GPSR a step further and presented an algorithm called "Adaptive Face Routing" (AFR). AFR is guaranteed to achieve a worst case cost of $O(c^2)$, if c is the cost of the best path. Like GPSR, AFR combines greedy routing with face routing to ensure routing convergence without compromising performance. As we have seen in Section 3.6, since the worst-case cost for any localized position-based routing algorithm is $\Omega(c^2)$, AFR is asymptotically optimal. Following this work, authors presented another algorithm, GOAFR$^+$ (pronounced as "gopher-plus") that improves performance in random graphs, while maintaining the asymptotically optimal properties of AFR [37].

5 Pre-processing Algorithms for Wireless *Ad Hoc* Networks

Before nodes can run any routing algorithm, they need to apply a pre-processing algorithm to create some kind of underlying graph. Usually, nodes start by exchanging beacon messages to create an initial connectivity graph (e.g., UDG). However, the properties of this initial graph may not be adequate for all routing algorithms. For instance, right-hand and hybrid routing algorithms, like FACE-1 or GPSR, need to use an underlying planar graph. This requires the pre-processing algorithm to remove some of the existing edges in a process known as "topology control". Since the details of the pre-processing algorithms widely differ between wireless and wired networks, we separate the presentation of the two cases. In this section we focus on the wireless case, while we defer the wired case to the next section.

A wireless *ad hoc* network is comprised of nodes that run on batteries and communicate with each other directly via radio using wireless links. A distinctive feature of wireless *ad hoc* networks is the lack of a fixed infrastructure of support. As a consequence, nodes must self-organize in a decentralized way. These characteristics turn these networks eligible for use in a number of circumstances, including the following:

- casual meetings, like conferences or sports events;
- catastrophic situations, where fixed infrastructures are no longer running;
- for rescue or military teams in inaccessible or hostile regions;
- self-managed networks of sensors, where new sensors can be added or sensors can go down due to battery exhaustion, at any moment. These networks may be used to monitor the environment (e.g., to detect fire or flooding), for health, home or commercial applications.

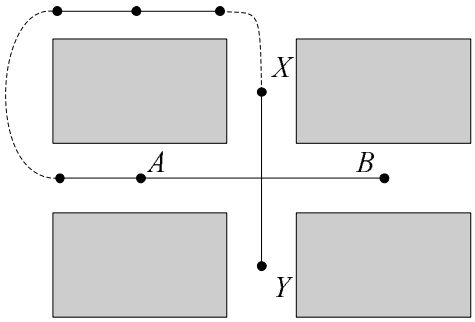

Figure 8: The number of hops between A and X may be arbitrarily large

Wireless *ad hoc* networks can be further characterized by having few resources, both at the node and at the network level. Nodes have limited memory, processing power and possibly worst of all, short batteries. On the other hand, the existing network bandwidth must be divided by all the nodes within reach. Additionally, topology tends to change very fast and radio communication is typically very expensive in terms of energy. Therefore, there is a strong motivation to use simple and economical pre-processing algorithms.

We assume that nodes have already exchanged beacon messages to know of each other and as a result they have already formed an initial graph according to the UDG model. The goal of the following pre-processing algorithms is to create a final planar graph. With the exception of the Gabriel graph that we present in the next section, all the other algorithms are restricted to the UDG model. In fact, eliminating intersecting edges in more general models is a quite difficult problem. If the connectivity between nodes is arbitrary (i.e., if it is not constrained to any function of distance), there is no localized algorithm capable of ensuring that all intersections are removed from the graph. This can be seen in Figure 8, where the intersection between edges AB and XY cannot be eliminated in a localized fashion.

5.1 Gabriel Graph

Consider nodes A and B from the initial node set V. If circle whose diameter is AB, i.e., $d(A, B)$ is empty of other nodes from V, then edge AB belongs to the Gabriel graph. This is depicted in Figure 10, where the gray color shows the area that must be empty of nodes. The set of all Gabriel edges defines the Gabriel graph (GG). The GG is a subgraph of the DT. This property directly results from the empty circle property of the DT. In our context, we are more interested in the constrained GG, which only has edges with length at most 1. In general, we will always mean *constrained Gabriel graph* even when we do not use the word *constrained*. The constrained GG is a subgraph of UDG.

Although it is quite easy to see that in the UDG model it is possible to create the GG, Barrière *et al.* [23] proved that with the $R/r \leq \sqrt{2}$ model it is also possible to create the GG. Here, we only show why is it always possible to remove the intersections from the GG in a localized fashion. Assume that two edges AC and BD intersect as depicted in figure 9. At least one of the angles of the parallelogram $ABCD$ must be greater or equal to $\pi/2$. Without loss of generality, assume that $\angle ABC \geq \pi/2$. Clearly, $B \in d(A, C)$ and hence neither A nor C are aware of B or else AC would not be a Gabriel edge. Hence, $\|AB\| > r \wedge \|BC\| > r$. This implies that $\|AC\| > \sqrt{2} \cdot r \geq R$. This means that A and C could not see each other which is a contradicition. Therefore, graph is planar if $R/r \leq \sqrt{2}$.

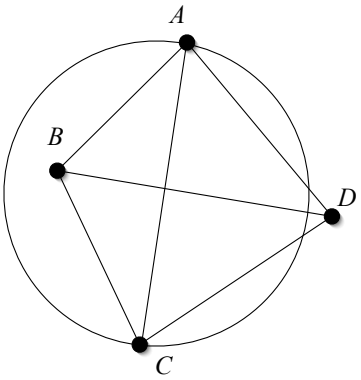

Figure 9: This intersection is impossible if $R/r \leq \sqrt{2}$

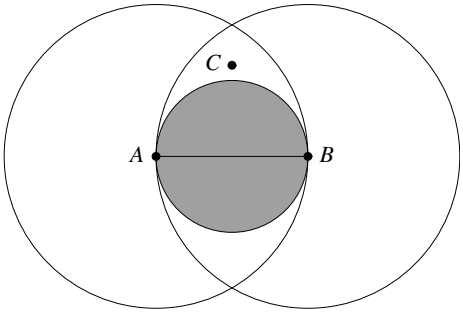

Figure 10: In a GG, the gray area must be clear

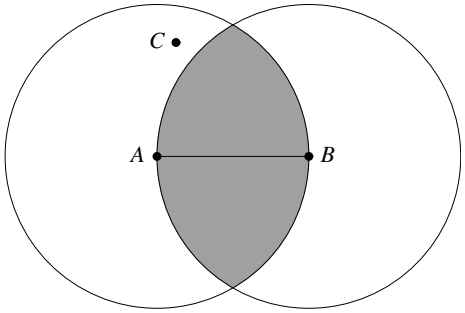

Figure 11: In a RNG, the gray area must be clear

5.2 Relative Neighborhood Graph

In the relative neighborhood graph (RNG), an edge AB exists when there is no third node $C \in V$ such that edges AC and BC are both shorter than AB. Again, in our context, we are interested in the constrained RNG graph, which only has edges with length at most 1. Both GG and RNG are planar. Both are connected as long as initial graph is also connected. Figure 11 shows the gray area that must be free of nodes. The "exclusion zone" of the RNG contains the corresponding zone of the GG, which means that the restricted RNG is a subgraph of the restricted GG and of the UDG. The RNG is also a subgraph of the DT.

5.3 Delaunay Triangulations for Wireless Networks

The problem with both GG and RNG is that neither one of them is a good spanner of the initial connection graph [38], which means that, in particular, they are not good spanners of UDG. One way to create better spanner graphs is to use a triangulation. However, under the UDG model, a complete Delaunay triangulation may not exist, because some edges may be longer than 1. Furthermore, even if all the Delaunay edges were shorter than 1, creating a Delaunay triangulation would make a routing scheme fail the criteria of being localized. This happens because ensuring the empty circumcircle property may require information of nodes that may be close in terms of Euclidean distance, but that may be very far away in terms of number of hops. Even though, this triangulation still owns some attractive properties. For instance, if nodes have similar views of their neighborhood, they can deterministically compute the same triangulation. This may save many steps to reach a form of agreement among the nodes. Additionally, it is possible to create variants of the Delaunay triangulation that are good spanners of UDG. Next, we review some of

Algorithm 3 Algorithm that creates RDG

```
E(A) ← {AB|AB ∈ UDel(A)}
for all edge AB ∈ E(A) do
  for all C ∈ N(A) do
    if A, B ∈ N(C) and AB ∉ UDel(C) then
      Delete edge AB from E(A)
    end if
  end for
end for
```

these graphs.

Localized Delaunay Triangulations

The most obvious variation of the Delaunay triangulation is probably the Unit Delaunay triangulation ($UDel$). $UDel$ results from the intersection of the Delaunay triangulation with the UDG graph. $UDel(V) = Del(V) \cap UDG(V)$, which means that $UDel$ is the Delaunay triangulation with edges that have length at most 1 [3]. $UDel$ is still impossible to build in a localized fashion and therefore, we will use another definition proposed in [27] of "k-localized Delaunay graph over a node set V", $LDel^{(k)}(V)$. $LDel^{(k)}(V)$ is comprised of two types of edges (not longer than 1):

- all edges from the GG;
- edges of all triangles ABC for which there are no nodes inside $\bigcirc ABC$ reachable by A, B or C in k or fewer hops.

Li *et al.* [27] proved that $LDel^{(k)}(V)$ is planar for $k \geq 2$, but edges may intersect for $k = 1$.

$PLDel(V)$ [27, 30], which stands for "Planar Localized Delaunay Triangulation", is defined as a planar graph comprised of all triangles of $LDel^{(1)}(V)$, *except* intersecting triangles that do not belong to $LDel^{(2)}(V)$. Li *et al.* [27] proved that $UDel(V)$ is a $(4\sqrt{3}\pi)/9$-spanner of $UDG(V)$ and that $LDel^{(k)}(V) \supseteq UDel(V)$. Hence $PLDel(V)$ and $LDel^{(k)}(V)$, for all k, are also $(4\sqrt{3}\pi)/9$-spanners of $UDG(V)$.

Restricted Delaunay Graph (RDG)

Gao *et al.* [28] presented an algorithm that creates a graph called the Restricted Delaunay Graph (RDG). RDG is simply any planar super-graph of $UDel$. Hence, $RDG(V) \supseteq UDel(V)$, which means that RDG is also a $(4\sqrt{3}\pi)/9$-spanner of $UDG(V)$. Their RDG graph is relatively simple, although possibly expensive in terms of communication. After exchanging information of its own position with its neighbors, each node uses an additional communication step to broadcast its own view of the Delaunay triangulation. This serves to make the triangulation between the nodes converge and to eliminate possible intersections. Let $UDel(A)$ be the Unit Delaunay triangulation computed by node A. Algorithm 3 shows the pseudo-code that the nodes must execute after this final communication step. Consider that node A executes this algorithm. Basically, A deletes edge AB if there is some node C that simultaneously knows A and B and that does not include AB in its triangulation. This means that AB is not a Delaunay edge.

Algorithms that create $PLDel(V)$

While the RDG is a good spanner of UDG (it is also a $(4\sqrt{3}\pi)/9$-spanner), its communication cost is $O(n^2)$, because each of the n nodes may see $O(n)$ nodes and $O(n)$ edges in the Delaunay

[3] Other possible name for this would be the "constrained Delaunay triangulation".

Algorithm 4 Algorithm that creates $LDel^{(1)}(V)$

Node A computes its Delaunay triangulation, $Del(N(A))$ with the neighbors that it is aware of, including itself
for all Edge $AB \in Del(N(A))$ **do**
  if AB is a Gabriel edge **then**
    Mark AB as final
    {AB will never be deleted}
  end if
end for
for all Triangles $\triangle ABC \in Del(N(A)) \mid ||AB|| \leq 1 \wedge ||AC|| \leq 1 \wedge ||BC|| \leq 1$ **do**
  {Ignore any triangle that has one or more long edges}
  if $\angle BAC \geq \pi/3$ **then**
    Broadcast a message proposal(A, B, C) to form a 1-localized Delaunay triangle $\triangle ABC \in LDel^{(1)}(V)$
  end if
end for
Receive messages from the other nodes
for all Message proposal(A, B, C) received **do**
  if $\triangle ABC \in Del(N(A))$ **then**
    {Accept the triangle}
    Broadcast accept(A, B, C)
  else
    {This triangle does not exist. Reject it}
    Broadcast reject(A, B, C)
  end if
end for
for all Triangles $\triangle ABC \in Del(N(A))$ **do**
  if B and C sent accept(A, B, C) or proposal(A, B, C) **then**
    Add edges AB and AC to the triangulation
  end if
end for

triangulation. To overcome this problem Li *et al.* [27] proposed an algorithm where nodes only announce *some* of their own edges. Especially in dense networks, this is considerably more economical. This algorithm builds $PLDel(V)$, which is, in fact, just a concrete case of an RDG. To simplify the presentation, we will follow the approach of the authors and maintain two separate parts, ran at node A. The first part, Algorithm 4, builds $LDel^{(1)}(V)$. We do not include a first step necessary to broadcast the identification and position of a given node. We assume that nodes already know the location of their neighbors. Although $LDel^{(1)}(V)$ is already a $(4\sqrt{3}\pi)/9$-spanner of UDG, it is not planar and, therefore, authors need Algorithm 5 to planarize $LDel^{(1)}(V)$. This algorithm removes all triangles that are intersected and that do not belong to $LDel^{(2)}(V)$.

First thing to notice is that as soon as nodes know the position of their neighbors, marking the Gabriel edges is costless in terms of communication. Hence, communication cost comes from announcing the triangulation. When node A computes $Del(N(A))$ it can have an arbitrarily large number of wrong triangles, i.e., triangles like $\triangle ABC$ that B and C know not to exist. If A were to announce all these triangles, communication cost could grow to $O(n^2 \log n)$. To avoid this, node A only announces the triangles for which $\angle BAC \geq \pi/3$. As a result, nodes can only announce up to 6 triangles in a first step, thus limiting the communication cost to $O(n \log n)$. Hence, there are some triangles that are not announced by A. However, this is not a problem, because at least in one of the vertexes, the angle must be greater than $\pi/3$. As a result, the other two nodes are forced to agree or disagree on the triangle (all without compromising the theoretical communication cost). In the end of this algorithm, a triangle $\triangle ABC$ can only exist if the three nodes have included it in their localized Delaunay triangulations ($Del(N(A))$ in the case of A). Hence, Algorithm 4 creates $LDel^{(1)}(V)$. Now, node A needs to ensure that it gets a planar graph, so it executes Algorithm 5. This algorithm uses two steps of communication first to remove possible intersections and then to check the edges that remain in the graph.

Besides [27], there is the work of Lan and Wen-Jing [30] that also builds $PLDel(V)$. However, although not precisely stated, the communication cost of their algorithm should raise to $O(n^2)$ if no optimizations are used. Figure 12 summarizes the relations between well-known graphs,

Algorithm 5 Algorithm that planarizes $LDel^{(1)}(V)$

Broadcast Gabriel edges incident on A and the triangles $\triangle ABC \in LDel^{(1)}(V)$
Receive the messages from other nodes
{Assume that A gathered information of triangles and Gabriel edges from all its neighbors}
for all Intersecting triangles $\triangle ABC$ and $\triangle XYZ$ known by A **do**
  if X, Y or $Z \in \bigcirc ABC$ **then**
    Remove triangle $\triangle ABC$
  end if
end for
Broadcast Gabriel edges and triangles $\triangle ABC$ incident on A that were not removed in the previous step
Listen to the messages from other nodes
for all Edge AB **do**
  if AB is a Gabriel edge or $\exists \triangle ABC | A, B, C$ announced $\triangle ABC$ in the previous step **then**
    Keep edge AB
  end if
end for

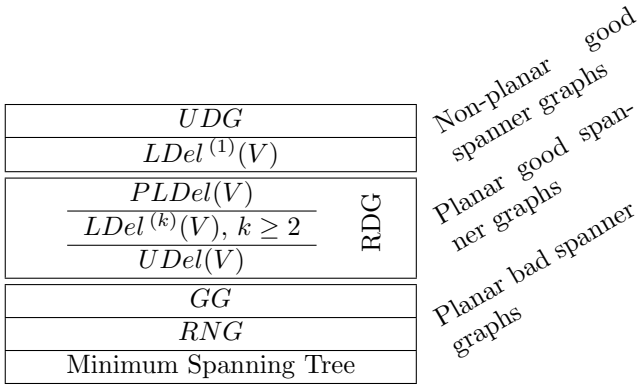

Figure 12: Relation between some well-known graphs

including these triangulations. Being on top of another graph means to contain such graph.

6 Pre-processing Algorithms for Wired Networks

Position-based routing can also be used in wired networks to enrich wired applications with geographically limited broadcasts or queries. Some authors, like Liebeherr *et al.* [39] use position to support multicast. Recent work shows that even in wired networks, the geographical position of the nodes reflects the topology of the network. [17] shows that in the Internet there is a strong correlation between physical and topological distances. Furthermore, authors believe that this correlation will increase as the Internet is expected to become more richly connected. Hence, the use of position may become increasingly important as a mean to route messages even in wired networks.

However, when applied to wired networks, position-based routing uses different assumptions. To start with, wired nodes and networks have typically much more resources than their wireless counterparts. Another important difference is that more often than not, wired networks do not have a communication range that looks like a circle, nor do they have easy access to a broadcast communication channel. As a consequence, the UDG model makes little sense and so does the notion of localized routing scheme. This implies a shift in the concerns. In a wired network, nodes can do more than using the pre-processing algorithms presented in Section 5. In fact, nodes can use more elaborate, non-localized pre-processing algorithms. For example, Liebeherr *et al.* [39] creates a complete Delaunay triangulation to support multicast at the application layer. Dobkin *et*

Table 1: Comparison of pre-processing algorithms for wireless and wired networks

	RNG	GG	[28]	$PLDel(V)$	$LDel^{(k)}(V)$, $k \geq 2$	DT
Wireless	S	S	S	S	P	I
Wired	P	P	N	N	N	P

al. [40] showed that the Delaunay triangulation is a $(1+\sqrt{5})\pi/2$-spanner of the complete Euclidean graph. This bound was later improved in [27] to $(4\sqrt{3}\pi)/9$, which is a smaller number ($\approxeq 2.42$).

A non-localized pre-processing algorithm offers some important advantages to the routing algorithm. In the case of a complete Delaunay triangulation, it is no longer necessary to use an algorithm like GPSR [36], because Bose and Morin [26, 41] showed that both the Greedy and Compass converge in a Delaunay triangulation (however, they can be defeated by random triangulations). Another advantage is that, in wired networks, it is reasonable to build a c-competitive (non-localized) routing scheme. Bose and Morin [26] presented an $O(1)$ routing algorithm called "Parallel Voronoi Routing" that is c-competitive in Delaunay triangulations. Later, they extended this result for a broader class of triangulations [41]. Unfortunately, the same authors have also shown that, in practice, performance of Parallel Voronoi Routing is worse than that of greedy or compass routing [26]. Therefore, although these latter algorithms can achieve poor results in some pathological cases, in the normal random case their performance is satisfactory. For this reason, in wired networks, we advocate the use of a simpler algorithm, such as the greedy routing algorithm.

7 Comparison of Routing Schemes

7.1 Comparison of Pre-processing Algorithms

As a result of the differences between wired and wireless networks, the kind of graph that is more appropriate to each environment varies. For instance, in wireless environments it is crucial to use few messages and to use localized pre-processing algorithms. Unlike this, in wired environments there is no broadcasting facility and a higher communication overhead is admissible. In Table 1, we evaluate the difficulty of creating each of the graphs presented in the previous sections, for wired and wireless environments. We use the following abbreviations: "S" — possible and simple; "P" — possible but may take several rounds of messages or have a high communication cost; "I" — impossible; "N" — makes little sense. Some of the entries of the table are to some extent subjective. Hence, according to this logic, all algorithms take several rounds in wired networks (when compared to wireless networks). We also classify the $LDel^{(k)}(V)$, $k \geq 2$ algorithm for wireless networks with a "P", because good algorithms to do this graph tend to be more complex than other triangulations [42, 43]. Naturally, new, simpler algorithms could make this classification change. Finally, it is to some extent pointless to create most of the triangulations in a wired environment, because it is not obvious whether those would be simpler than creating the complete DT.

7.2 Comparison of Routing Algorithms

Since the main goal of any routing algorithm is to deliver packets, the most important evaluation criterion is unquestionably the delivery success rate. However, guaranteed delivery cannot be achieved at any price, for instance, by flooding each packet throughout the entire network. Hence, another relevant criterion is the communication effort of the algorithm. In particular, one important distinction is whether the algorithm uses flooding to deliver packets or not. Memory requirement of the algorithm is also important, because some algorithms require information about

Table 2: Comparison of routing algorithms

Algorithm	Guarant. del.	Local.	Memory	c-comp.
Greedy, Compass	No/DT	Yes/No	Memoryless	No
MFR	No	Yes	Memoryless	No
Rand. compass	Yes	Yes	Memoryless	No
GEDIR	No/DT	Yes/No	$O(1)$	No
Voronoi	*DT*	No	$O(1)$	No
Parallel Voronoi	*DT*	No	$O(1)$	Yes
FACE-1, FACE-2, GPSR, AFR, GOAFR$^+$	Planar G.	Yes	$O(1)$	No
SP Algorithms	Yes	No	$O(n \log n)$, $O(1)$	Yes

past packets or require nodes to store large routing tables. It is also important to know if the algorithm is localized, according to the definition of Section 3.4.

Table 2 resumes a comparison between routing algorithms. We include "Shortest Path Algorithms" (SP algorithms), i.e., algorithms that are not based on position. These algorithms inherit from the techniques of the wired IP networks. The values of the table are valid for the Destination-Sequenced Distance-Vector (DSDV) [9]; *Ad hoc* On-Demand Distance Vector Routing (AODV) [10] and Dynamic Source Routing (DSR) [11]. We assumed a worst-case scenario where all nodes have routing entries to all other nodes, thus requiring $O(n \log n)$ memory, per node, where n is the number of nodes. In general, this upper bound is unreachable. The memory entry of the table for SP algorithms includes this value, followed by the space required by the algorithm in each packet. Position-based algorithms only include the latter, because space needed to store the graph strongly depends on the pre-processing algorithm used (but this is typically $O(\log n)$). Regarding the space used in each packet, all the algorithms need at least $O(\log n)$ bits to identify the destination. However, to make a clearer distinction between them, in Table 2, we refer to the number of other nodes in the packet, besides the destination ($O(1)$ means that the algorithm needs the source, while *memoryless* means that the source is not needed). Of the routing algorithms included in the table, Greedy, MFR, GEDIR, Randomized Compass, Voronoi as well as shortest path algorithms are loop-free. The face and hybrid algorithms are also loop free in the sense that partial loops occur in a controlled way, in planar graphs.

Of the localized algorithms included, only randomized compass guarantees delivery in arbitrary graphs. Some algorithms guarantee delivery if the graph is a Delaunay triangulation, while face algorithms guarantee delivery if the underlying graph is planar. Remaining localized algorithms do not guarantee delivery. Of the position-based algorithms, only parallel Voronoi is c-competitive (see section 3.6). We assume that SP algorithms can find the optimal path, under favorable circumstances. Algorithms may use hop count, energy or distance as their metric. Finally, of the algorithms that we include in the comparison, only shortest path algorithms use flooding. Depending on the particular case, these algorithms use flooding to propagate route requests or to propagate routing tables.

8 Cluster-Based Algorithms

Clustering a network consists of dividing that network into groups of nodes. Usually, each cluster will have a "cluster-head" that will act as the representative of that group of nodes. In a sense, a cluster (often by means of its cluster-head) will act as a kind of super-node that represents all the nodes of the group. This creates a network that is much sparser. As a consequence, the man-

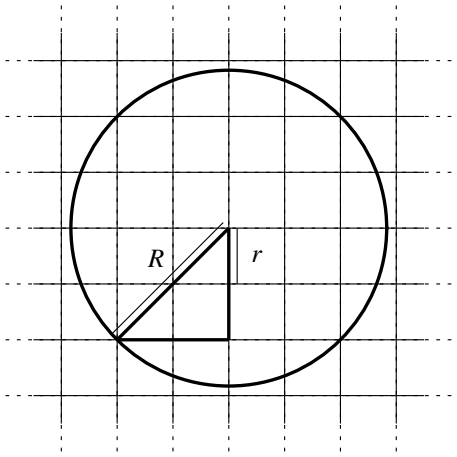

Figure 13: Division of the space into cells of fixed size

agement of position and non-position-based routing algorithms becomes much simpler and most nodes send fewer control packets, thus reducing collisions and battery consumption. The drawback of clustering is that, often, some unlucky nodes will have more service than others. Additionally, a sparser network with fewer nodes, may reduce the routing options for most algorithms. In the case of position-based routing algorithms, Greedy, MFR, GEDIR, Compass, Random compass, among many others, should have worse behavior on sparser networks, because they have fewer options there. On the contrary, face and hybrid routing algorithms benefit from clustering. Since they operate in planar graphs, the sparser the graph is (only in terms of nodes, not in terms of edges), the longer are the edges. Longer edges will result in fewer hops to reach destination. The algorithm GOAFR$^+$ [37] explores this idea. This result is also clearly shown in [44].

8.1 Division into cells

There is a very large body of work that focus on clustering [e.g. 45, 46, 47, 28, 29, 48, 49, 50, 51, 52, 53, 54]. We only present a very simple clustering technique of Xu *et al.* [55], because it is based on positional information and it can be used in several contexts. Xu *et al.* [55] use this technique in the "Geographical Adaptive Fidelity" (GAF) algorithm to conserve energy. GAF divides the space into equally-sized cells that have the shape of squares (assuming that the entire space is a square), as the grid depicted in Figure 13. Division of the space into cells allows a simple definition of the network of clusters: all nodes inside the same cell belong to the same cluster. The size of the cells is limited by the communication range of the nodes, because nodes in a cell must always listen to any other node either in its own cell or in any adjacent cell. This restriction ensures that in most circumstances, the clustered network stays connected, as long as the initial network is also connected. If we assume that nodes have a communication range of R, the resulting square side is at most $R/\sqrt{5}$ or $R/\sqrt{8}$, if 4 or 8 cells surrounding the node are considered to be adjacent, respectively. The goal of GAF is to put as many nodes as possible to sleep and maintain only one node active in each cluster. After a predetermined period of time, sleeping nodes must wake up to substitute active nodes. This clustering technique can also be used to create a distributed hash table for wireless *ad hoc* networks [56, 44]. A similar clustering technique can also be found in [57].

9 Summary

In this paper, we surveyed position-based routing schemes. A routing scheme is comprised of a pre-processing algorithm plus a routing algorithm. In wireless networks, a popular routing scheme consists of combining the greedy perimeter stateless routing algorithm (GPSR) with a planar graph, like the Gabriel graph. Triangulations can replace the Gabriel graph to achieve better performance, but they are much more difficult to create. In wired networks, position-based routing is a simpler problem, because nodes can usually afford to create a complete Delaunay triangulation. This enables the use of routing algorithms even simpler than GPSR, such as greedy.

References

[1] F. Araújo, "Positin-based distributed hash tables," Ph.D. dissertation, University of Lisbon, Lisbon, Portugal, 2005, (to appear).

[2] P. Fraigniaud and C. Gavoille, "A space lower bound for routing in trees," in 19^{th} *Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, vol. Lecture Notes in Computer Science. Springer, 2002. [Online]. Available: citeseer.nj.nec.com/470210.html

[3] J. van Leeuwen and R. Tan, "Compact routing methods: A survey," Universiteit Utrecht, Tech. Rep. UU-CS-1995-05, 1995.

[4] Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, pp. 269–271, 1959.

[5] R. Bellman, *Dynamic Programming*. Princeton, N.J.: Princeton University Press, 1957.

[6] L. F. Jr. and D. Fulkerson, *Flows in Networks*. Princeton, N.J.: Princeton University Press, 1962.

[7] R. Jain, A. Puri, and R. Sengupta, "Geographical routing using partial information for wireless ad hoc networks," 1999. [Online]. Available: citeseer.nj.nec.com/jain99geographical.html

[8] J. Li, J. Jannotti, D. De Couto, D. Karger, and R. Morris, "A scalable location service for geographic ad-hoc routing," in *Proceedings of the 6th ACM International Conference on Mobile Computing and Networking (MobiCom '00)*, August 2000, pp. 120–130. [Online]. Available: citeseer.nj.nec.com/li00scalable.html

[9] C. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," in *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, 1994, pp. 234–244. [Online]. Available: citeseer.nj.nec.com/article/perkins94highly.html

[10] C. Perkins, "Ad-hoc on-demand distance vector routing," 1997. [Online]. Available: citeseer.nj.nec.com/article/perkins97adhoc.html

[11] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*, Imielinski and Korth, Eds. Kluwer Academic Publishers, 1996, vol. 353. [Online]. Available: citeseer.nj.nec.com/johnson96dynamic.html

[12] I. Stojmenovic, "Position-based routing in ad hoc networks," *IEEE Communications Magazine*, July 2002.

[13] A. Rao, C. Papadimitriou, S. Shenker, and I. Stoica, "Geographic routing without location information," in *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM Press, 2003, pp. 96–108.

[14] M. Wattenhofer, R. Wattenhofer, and P. Widmayer, "Geometric routing without geometry," in *12th Colloquium on Structural Information and Communication Complexity.* Mont-St-Michel, France: Springer-Verlag, LNCS 3499, May 2005.

[15] D. Niculescu and B. Nath, "Vor base stations for indoor 802.11 positioning," in *MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking.* ACM Press, 2004, pp. 58–69.

[16] A. Haeberlen, E. Flannery, A. M. Ladd, A. Rudys, D. S. Wallach, and L. E. Kavraki, "Practical robust localization over large-scale 802.11 wireless networks," in *MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking.* ACM Press, 2004, pp. 70–84.

[17] V. N. Padmanabhan and L. Subramanian, "An investigation of geographic mapping techniques for internet hosts," *SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 4, pp. 173–185, 2001.

[18] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker, "GHT: A geographic hash table for data-centric storage in sensornets," in *First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, Atlanta, Georgia, September 2002.

[19] A. Cerpa, N. Busek, , and D. Estrin, "Scale: A tool for simple connectivity assessment in lossy environments," UCLA Center for Embedded Network Sensing (CENS), Tech. Rep. 0021, September 2003.

[20] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker, "Complex behavior at scale: An experimental study of low-power wireless sensor networks," UCLA, Tech. Rep. CSD-TR 02-0013, February 2002.

[21] Y.-J. Kim, R. Govindan, B. Karp, and S. Shenker, "Geographic routing made practical," in *Second Symposium on Networked Systems Design and Implementation*, Boston, MA, USA, May 2005.

[22] M. H. Lim, A. Greenhalgh, J. Chesterfield, and J. Crowcroft, "Hybrid routing: A pragmatic approach to mitigating position uncertainty in geo-routing," University of Cambridge, Tech. Rep. UCAM-CL-TR-629, April 2005.

[23] L. Barrière, P. Fraigniaud, L. Narayanan, and J. Opatrny, "Robust position-based routing in wireless ad hoc networks with irregular transmission ranges," *Wireless Communications And Mobile Computing journal*, 2002. [Online]. Available: citeseer.nj.nec.com/492646.html

[24] E. Kranakis, H. Singh, and J. Urrutia, "Compass routing on geometric networks," in *11th Canadian Conference on Computation Geometry (CCCG 99)*, 1999.

[25] F. Kuhn, R. Wattenhofer, and A. Zollinger, "Asymptotically optimal geometric mobile ad-hoc routing," in *6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM'02)*, 2002.

[26] P. Bose and P. Morin, "Online routing in triangulations," in *10th Annual Internation Symposium on Algorithms and Computation (ISAAC)*, 1999.

[27] X.-Y. Li, G. Calinescu, and P.-J. Wan, "Distributed construction of a planar spanner and routing for ad hoc wireless networks," in *The 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2002.

[28] J. Gao, L. J. Guibas, J. Hershberger, L. Zhang, and A. Zhu, "Geometric spanners for routing in mobile networks," in *2nd ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 01)*, 2001.

[29] Y. Wang and X.-Y. Li, "Geometric spanners for wireless ad hoc networks," in *The 22nd IEEE International Conference on Distributed Computing Systems*, 2002.

[30] L. Lan and H. Wen-Jing, "Localized delaunay triangulation for topological construction and routing on manets," in *2nd ACM Workshop on Principles of Mobile Computing (POMC'02)*, 2002.

[31] G. G. Finn, "Routing and addressing problems in large metropolitan-scale internetworks," Institute for Scientific Information, Tech. Rep. ISU/RR-87-180, March 1987.

[32] I. Stojmenovic and X. Lin, "Loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, no. 10, October 2001.

[33] H. Takagi and L. Kleinrock, "Optimal transmission ranges for randomly distributed packet radio terminals," *IEEE Transactions on Communications*, vol. 32, no. 3, pp. 246–257, March 1984.

[34] J. A. Bondy and U. S. R. Murty, *Graph Teory with Applications.* Elsevier North-Holland, 1976.

[35] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, "Routing with guaranteed delivery in *ad hoc* wireless networks," in *International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM)*, 1999, pp. 48–55.

[36] B. Karp and H. T. Kung, "GPRS: Greedy perimeter stateless routing for wireless networks," in *ACM/IEEE International Conference on Mobile Computing and Networking*, 2000.

[37] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger, "Geometric ad-hoc routing: Of theory and practice," in *22nd ACM Symposium on the Principles of Distributed Computing (PODC 2003)*, Boston, Massachusetts, July 2003.

[38] D. Eppstein, "Spanning trees and spanners," in *Handbook of Computational Geometry.* Amsterdam: Elsevier North-Holland, 2000, pp. 425–461.

[39] J. Liebeherr, M. Nahas, and W. Si, "Application-layer multicasting with Delaunay triangulation overlays," University of Virginia, Department of Computer Science, Charlottesville, VA 22904, Tech. Rep. CS-2001-26, May 2001. [Online]. Available: citeseer.nj.nec.com/liebeherr01applicationlayer.html

[40] D. Dobkin, S. J. Friedman, and K. J. Supowit, "Delaunay graphs are almost as good as complete graphs," *Discrete Computational Geometry*, July 1990.

[41] P. Bose and P. Morin, "Competitive online routing in geometric graphs," in *8th Colloquium on Structural Information & Communication Complexity.* Carleton University Press, June 2001, pp. 35–44.

[42] G. Calinescu, "Computing 2-hop neighborhoods in ad hoc wireless networks," Adhoc-Now '03, 2003.

[43] Y. Wang and X.-Y. Li, "Localized construction of bounded degree and planar spanner for wireless ad hoc networks," in *DIALM-POMC '03: Proceedings of the 2003 joint workshop on Foundations of mobile computing.* New York, NY, USA: ACM Press, 2003, pp. 59–68.

[44] F. Araújo, L. Rodrigues, J. Kaiser, C. Liu, and C. Mitidieri, "Chr: a distributed hash table for wireless ad hoc networks," in *The 25th IEEE International Conference on Distributed Computing Systems Workshops (DEBS '02)*, Columbus, Ohio, USA, June 2005.

[45] S. Ni, Y. Tseng, Y. Chen, and J. Sheu, "The broadcast storm problem in a mobile ad hoc network," in *Conference on Mobile Computing, MOBICOM*, August 1999, pp. 151–162.

[46] B. Das and V. Bharghavan, "Routing in ad-hoc networks using minimum connected dominating sets," in *ICC (1)*, 1997, pp. 376–380. [Online]. Available: citeseer.nj.nec.com/article/das97routing.html

[47] C.R.Lin and M. Gerla, "Adaptive clustering for mobile wireless networks," *IEEE Journal Selected Areas in Communication*, vol. 15, no. 7, pp. 1265–1275, 1997.

[48] G. Chen and I. Stojmenovic, "Clustering and routing in wireless ad hoc networks," Department of Computer Science, SITE, University of Ottawa, Ottawa, Ontario K1N 6N5, Canada, Tech. Rep. TR-99-05, June 1999.

[49] *Radio Equipment and Systems: High Performance Radio Local Area Network Type 1, Functional Specifications*, ITSI STC-RES10 Committee, June 1996.

[50] G. Calinescu, I. Mandoiu, P. Wan, and A. Zelikovsky, "Selecting forwarding neighbors in wireless ad hoc networks," in *Fifth International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communication (DIALM)*, 2001.

[51] A. Qayyum, L. Viennot, and A. Laouiti, "Multipoint relaying: An efficient technique for flooding in mobile wireless networks," INRIA, Tech. Rep. Research Report RR-3898, March 2000.

[52] J. Wu and H. Li, "A dominating set based routing scheme in ad hoc wireless networks," in *Third International Workshop Discrete Algorithms and Methods for Mobile Computing and Communication (DIALM)*, August 1999, pp. 7–14.

[53] I. Stojmenovic, M. Seddigh, and J. Zunic, "Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 1, pp. 14–25, January 2002.

[54] S. Ni, Y. Tseng, and J. Sheu, "Efficient broadcasting in a mobile ad hoc network," in *International Conference on Distributed Computing and Systems (ICDCS'01)*, April 2001, pp. 16–19.

[55] Y. Xu, J. S. Heidemann, and D. Estrin, "Geography-informed energy conservation for ad hoc routing," in *Mobile Computing and Networking*, 2001, pp. 70–84. [Online]. Available: citeseer.nj.nec.com/xu01geographyinformed.html

[56] M. Li, W.-C. Lee, and A. Sivasubramaniam, "Efficient peer-to-peer information sharing over mobile ad hoc networks," in *Second Workshop on Emerging Applications for Wireless and Mobile Access (MobEA II), in conjunction with the World Wide Web Conference (WWW)*, May 2004.

[57] I. Gupta, R. van Renesse, and K. P. Birman, "Scalable fault-tolerant aggregation in large process groups," in *DSN '01: Proceedings of the 2001 International Conference on Dependable Systems and Networks (formerly: FTCS)*. IEEE Computer Society, 2001, pp. 433–442.